

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-215080

(43)Date of publication of application : 04.08.2000

(51)Int.Cl.

G06F 11/28
G06F 15/00
G06F 15/16
G06F 15/177

(21)Application number : 11-013490

(71)Applicant : NTT COMMUNICATIONWARE
CORP

(22)Date of filing : 21.01.1999

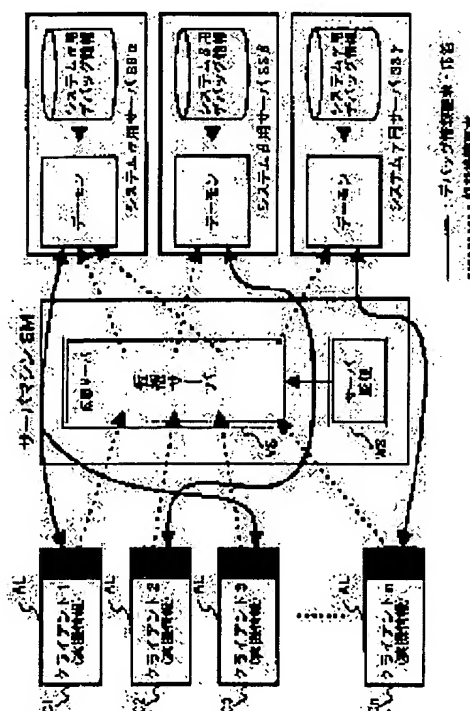
(72)Inventor : HIRANO MITSUNORI
YAMADA YOICHI
KAMIHITO MASAOKI
TATE TAKESHI

(54) TERMINAL DEVICE, REPEATING DEVICE, SERVER AND DEBUGGING
INFORMATION ACQUISITION SYSTEM, DEBUG INFORMATION ACQUIRING METHOD,
AND RECORDING MEDIUM

(57)Abstract:

PROBLEM TO BE SOLVED: To obtain a terminal device, a repeating device, server and debug information acquisition system, debug information acquiring method, and recording medium which can obtain desired symbol information remotely without paying attention to differences of a target debug machine.

SOLUTION: When a client C specifies a desired system, the specified system is connected to a system-adaptive server SS accessible to the debug information of the desired system through a virtual server VS and at a symbol conversion request for a program to be debugged, symbol information is sent from the system-adaptive server SS to the client C.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-215080

(P2000-215080A)

(43) 公開日 平成12年8月4日(2000.8.4)

(51) Int.Cl. ⁷	識別記号	F I	テームト* (参考)
G 0 6 F 11/28		G 0 6 F 11/28	L 5 B 0 4 2
15/00	3 2 0	15/00	3 2 0 J 5 B 0 4 5
15/16	6 2 0	15/16	6 2 0 B 5 B 0 8 5
15/177	6 7 8	15/177	6 7 8 H

審査請求 未請求 請求項の数 8 O L (全 17 頁)

(21) 出願番号 特願平11-13490
 (22) 出願日 平成11年1月21日(1999.1.21)

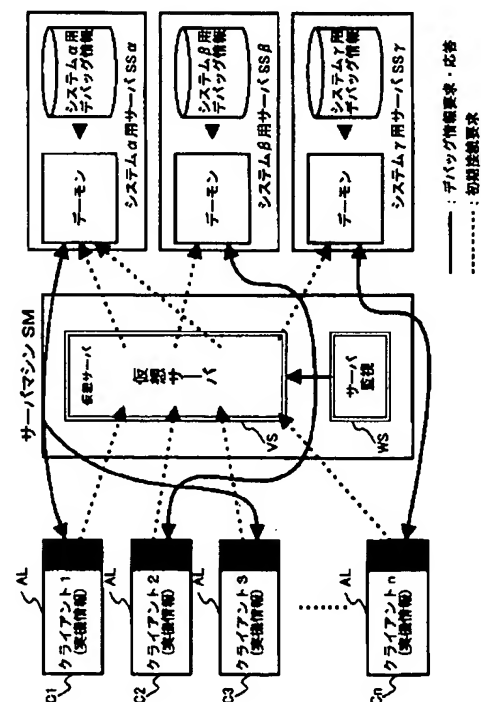
(71) 出願人 397065480
 エヌ・ティ・ティ・コミュニケーションウ
 ェア株式会社
 東京都港区港南一丁目9番1号
 (72) 発明者 平野 光徳
 東京都港区港南一丁目9番1号 エヌ・テ
 ィ・ティ・コミュニケーションウエ株式
 会社内
 (74) 代理人 100098084
 弁理士 川▲崎▼ 研二

最終頁に続く

(54) 【発明の名称】 端末装置、中継装置、サーバならびにデバッグ情報取得システム、デバッグ情報取得方法および
 (57) 【要約】 記録媒体

【課題】 ターゲットとなるデバッグマシンの差異を意識することなく、遠隔で所望のシンボル情報を取得することができる、端末装置、中継装置、サーバならびにデバッグ情報取得システム、デバッグ情報取得方法および記録媒体を提供する。

【解決手段】 クライアントCから所望のシステムを指定すると、仮想サーバVSを介して所望のシステムのデバッグ情報にアクセス可能なシステム対応サーバSSと接続され、デバッグ対象プログラムに対するシンボル変換要求に応答して、システム対応サーバSSからクライアントCに向けてシンボル情報が送信される。



【特許請求の範囲】

【請求項1】 異なる複数のシステムのデバッグ情報の内、1または2以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバと中継装置を介して接続され、前記デバッグ情報の中から所望のシステムのデバッグ情報を取得してプログラムのデバッグが行われる端末装置であって、

回線の接続を要求する旨の信号を前記中継装置に送信する手段と、

この回線接続要求信号に応答して前記中継装置から送信された回線の接続を承認する旨の信号を受信した後、前記複数のシステムの中から所望のシステムを指定して前記中継装置に通知する手段と、

この指定されたシステムのデバッグ情報にアクセス可能なサーバと前記中継装置を介して接続された後、当該サーバに向けてデバッグ対象のプログラムに対するシンボル変換要求を発行する手段と、

このシンボル変換要求に応答して該サーバから送信されたデバッグ情報を受信する手段と、
を備えたことを特徴とする端末装置。

【請求項2】 端末装置と、異なる複数のシステムのデバッグ情報の内、1または2以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバとの間に介挿され、前記端末装置と前記サーバとを接続する中継装置であって、

前記端末装置からの回線接続要求に応答して回線接続の可否を前記端末装置に通知する手段と、

前記端末装置から指定されたシステムに関する通知を受け取る手段と、

この指定されたシステムのデバッグ情報にアクセス可能なサーバを前記複数のサーバの中から選出して当該サーバへの接続の可否を判定するとともに、該サーバへの接続が可能であると判定した場合に該サーバと前記端末装置とを接続する手段と、

を備えたことを特徴とする中継装置。

【請求項3】 中継装置を介して端末装置と接続され、1または2以上のシステムのデバッグ情報であってシンボル情報を示唆する多数の構成要素からなるデバッグ情報にアクセス可能なサーバであって、

前記端末装置から発行されたデバッグ対象のプログラムに対するシンボル変換の要求に応答して、前記デバッグ情報の中から前記シンボル変換を行うために必要な構成要素を抽出する抽出手段と、

この抽出された構成要素を用いてシンボル情報を作成し、前記端末装置に向けて送信する送信手段と、
を備えたことを特徴とするサーバ。

【請求項4】 請求項3に記載のサーバにおいて、前記端末装置から発行されたシンボル変換要求を所定のインタフェースを介して受け付ける手段を備え、前記送信手段は、この所定のインタフェースを介して、

作成されたシンボル情報を前記端末装置に向けて送信することを特徴とするサーバ。

【請求項5】 請求項3または4に記載のサーバにおいて、

前記端末装置からのシンボル変換要求に応じたシンボル情報を作成することができない場合には、所定の既定値が格納されたデータを前記端末装置に向けて送信することを特徴とするサーバ。

【請求項6】 請求項1に記載の端末装置と、請求項2に記載の中継装置と、請求項3～5のいずれかに記載のサーバとを備えたことを特徴とするデバッグ情報取得システム。

【請求項7】 異なる複数のシステムのデバッグ情報の内、1または2以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバのいずれかと端末装置とが中継装置を介して接続され、前記デバッグ情報の中から所望のシステムのデバッグ情報を前記端末装置側で取得するデバッグ情報取得方法であって、

回線の接続を要求する旨の信号を前記端末装置から前記中継装置に送信する過程と、

この回線接続要求信号に応答して前記中継装置から送信された回線の接続を承認する旨の信号を受信した前記端末装置が、前記複数のシステムの中から所望のシステムを指定して前記中継装置に通知する過程と、

この指定されたシステムのデバッグ情報にアクセス可能なサーバと前記中継装置を介して前記端末装置が接続された後、該端末装置から当該サーバに向けてデバッグ対象のプログラムに対するシンボル変換要求を発行する過程と、

このシンボル変換要求に応答して該サーバから送信されたデバッグ情報を前記端末装置側で受信する過程と、
を備えたことを特徴とするデバッグ情報取得方法。

【請求項8】 請求項7に記載の方法を実行するプログラムを記録したことを特徴とする記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】この発明は、シンボリック情報を取得してデバッグ作業を行う際に用いて好適な端末装置、中継装置、サーバならびにデバッグ情報取得システム、デバッグ情報取得方法および記録媒体に関する。

【0002】

【従来の技術】一般に、開発したソースプログラムの検証に際しては、該ソースプログラムをコンパイラによってコンパイルしてオブジェクトコードに変換し、これをデバッグに実行させることにより設計通りに動作するかどうかを確認する方法が採られる。このデバッグには、CやC++などの高級言語で書かれたプログラムをソースプログラムのレベルでデバッグできるシンボリックデバッグがある。このシンボリックデバッグでは、デバッグ対象のプログラムに基づき上記高級言語のコンパイラ

が生成・出力する情報（シンボル情報テーブル）やスタックフレーム等が利用され、ユーザがメモリ内容の参照と変更、プログラム動作のトレース、ブレークポイントの設定、変数の値設定、変数の値表示などを高水準言語のレベルで行い、デバッグ対象のプログラムの動作状況を随時確認することができる。

【0003】特に電話交換機の制御プログラムのような多岐に亘るプロセッサの下で稼働する大規模なソフトウェアの開発においては、このようなシンボリックデバッグはプロセッサ毎に異なるデバッグ情報を取得するために、複数のリモートサーバと複数の端末との間で行われる。図16は、このようなリモートサーバRS1、RS2、RS3とユーザ端末Tとからなるデバッグシステムの構成を模式的に示したものである。この場合、各サーバRS1～RS3はそれぞれ、システムα用のデバッグ情報、システムβ用のデバッグ情報、システムγ用のデバッグ情報を保有し、各端末を操作するユーザが所望のシステムのシンボル情報を取得してデバッグ作業が行われる。

【0004】

【発明が解決しようとする課題】しかし、このような従来のデバッグシステムでは、ターゲットとするシステムのデバッグマシンごと（図16に示した例ではシステムα、β、γのデバッグマシンごと）に、デバッグ情報およびそのデバッグ情報を利用するプログラムの両者を揃えることが必要であった。つまり、従来のデバッグ方式では、システムα用のデバッガ、システムβ用のデバッガ、・・・の如く個別にデバッガを作成する必要があった。従って、ターゲットマシンの種類が増えると、そのターゲットマシンにおけるデバッグ情報およびそのデバッグ情報を利用するためのプログラムの双方を揃えることが必要であった。

【0005】また、実際のデバッグ作業の際には、所望のデバッグ情報を保有するマシンにリモートでログインしなければならなかった。従って、ターゲットとなるマシンのアドレス（例えばIPアドレス）やホスト名、デバッグ情報ファイルやデバッグ情報を利用するプログラムの存在場所（ディスク内のパス名やファイル名）を予めユーザが認識している必要があった。そして、別のターゲットマシンに対してデバッグ作業を試みるには、ユーザが変更するマシンを意識しなければならないといった状況があった。さらに、特定のマシンに対するデバッグ作業に複数のユーザが集中すると、ホストマシンの負荷が増大し、デバッグ効率が悪くなるといった点も指摘されていた。

【0006】そこで、この発明は、このような従来の課題に着目してなされたもので、ターゲットとなるデバッグマシンの差異を意識することなく、遠隔端末から所望のシンボリック情報を取得してデバッグ作業を行うことができる端末装置、中継装置、サーバ、ならびにデバ

グ情報取得システム、デバッグ情報取得方法および記録媒体を提供することを目的とするものである。

【0007】

【課題を解決するための手段】上記目的を達成するために、第1発明の端末装置は、異なる複数のシステムのデバッグ情報の内、1または2以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバと中継装置を介して接続され、前記デバッグ情報の中から所望のシステムのデバッグ情報を取得してプログラムのデバッグが行われる端末装置であって、回線の接続を要求する旨の信号を前記中継装置に送信する手段と、この回線接続要求信号に応答して前記中継装置から送信された回線の接続を承認する旨の信号を受信した後、前記複数のシステムの中から所望のシステムを指定して前記中継装置に通知する手段と、この指定されたシステムのデバッグ情報にアクセス可能なサーバと前記中継装置を介して接続された後、当該サーバに向けてデバッグ対象のプログラムに対するシンボル変換要求を発行する手段と、このシンボル変換要求に応答して該サーバから送信されたデバッグ情報を受信する手段とを備えたことを特徴とするものである。

【0008】第2発明の中継装置は、端末装置と、異なる複数のシステムのデバッグ情報の内、1または2以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバとの間に介挿され、前記端末装置と前記サーバとを接続する中継装置であって、前記端末装置からの回線接続要求に応答して回線接続の可否を前記端末装置に通知する手段と、前記端末装置から指定されたシステムに関する通知を受け取る手段と、この指定されたシステムのデバッグ情報にアクセス可能なサーバを前記複数のサーバの中から選出して当該サーバへの接続の可否を判定するとともに、該サーバへの接続が可能であると判定した場合に該サーバと前記端末装置とを接続する手段とを備えたことを特徴とするものである。

【0009】第3発明のサーバは、中継装置を介して端末装置と接続され、1または2以上のシステムのデバッグ情報であってシンボル情報を示唆する多数の構成要素からなるデバッグ情報にアクセス可能なサーバであって、前記端末装置から発行されたデバッグ対象のプログラムに対するシンボル変換の要求に応答して、前記デバッグ情報の中から前記シンボル変換を行うために必要な構成要素を抽出する抽出手段と、この抽出された構成要素を用いてシンボル情報を作成し、前記端末装置に向けて送信する送信手段とを備えたことを特徴とするものである。

【0010】第4発明のサーバは、上記第3発明のサーバにおいて、前記端末装置から発行されたシンボル変換要求を所定のインタフェースを介して受け付ける手段を備え、前記送信手段は、この所定のインタフェースを介して、作成されたシンボル情報を前記端末装置に向けて

送信することを特徴とするものである。

【0011】第5発明のサーバは、上記第3または第4発明のサーバにおいて、前記端末装置からのシンボル変換要求に応じたシンボル情報を作成することができない場合には、所定の既定値が格納されたデータを前記端末装置に向けて送信することを特徴とするものである。

【0012】第6発明のデバッグ情報取得システムは、上記第1発明の端末装置と、上記第2発明の中継装置と、上記第3～第5発明のいずれかのサーバとを備えたことを特徴とするものである。

【0013】第7発明のデバッグ情報取得方法は、異なる複数のシステムのデバッグ情報の内、1または2以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバのいずれかと端末装置とが中継装置を介して接続され、前記デバッグ情報の中から所望のシステムのデバッグ情報を前記端末装置側で取得するデバッグ情報取得方法であって、回線の接続を要求する旨の信号を前記端末装置から前記中継装置に送信する過程と、この回線接続要求信号に応答して前記中継装置から送信された回線の接続を承認する旨の信号を受信した前記端末装置が、前記複数のシステムの中から所望のシステムを指定して前記中継装置に通知する過程と、この指定されたシステムのデバッグ情報にアクセス可能なサーバと前記中継装置を介して前記端末装置が接続された後、該端末装置から当該サーバに向けてデバッグ対象のプログラムに対するシンボル変換要求を発行する過程と、このシンボル変換要求に応答して該サーバから送信されたデバッグ情報を前記端末装置側で受信する過程とを備えたことを特徴とするものである。

【0014】第8発明の記録媒体は、上記第7発明の方法を実行するプログラムを記録したことを特徴とするものである。なお、このプログラムをCD-ROM、テープメディア、DVD-RAM等の記録媒体に格納して頒布する（ネットワーク経由でこのプログラムを配信したり、ダウンロードする行為も含む）ことによっても本発明を実施することができる。

【0015】

【発明の実施の形態】この発明の好ましい実施の形態について、以下、添付図面を参照しつつ詳細に説明する。

【0016】1. 実施形態の構成

1. 1. 基本構成

図1は、本実施形態にかかるデバッグシステムの概要を示したものであり、図2は、このデバッグシステムのより具体的な構成について示したものである。このデバッグシステムは、デバッグ用端末であるn台のクライアントC1～Cn（任意のクライアントを示す場合には「クライアントC」と表記するものとする）と、仮想サーバVSと監視用サーバWSとからなるサーバマシンSMと、それぞれ異なるCPU（ α , β , γ , ...）固有のデバッグ情報を保有するシステム対応サーバS

α , SS β , SS γ , ...（これらのシステム対応サーバの内、任意のシステム対応サーバを示す場合には「システム対応サーバSS」と表記するものとする）とで構成され、これらの装置は伝送路となるイーサネットENETによって相互に接続されている。

【0017】1. 2. クライアントおよびサーバマシンの構成

クライアントCは、ユーザがデバッグ作業を行うための端末装置である。このクライアントCからシステム対応サーバSSの記憶装置内に記憶されているソースプログラムを呼び出してデバッグ指令を与え、シンボル変換されたデバッグ情報（シンボリック情報）をシステム対応サーバSSから取得する。そして、このデバッグ情報を参照してソースプログラムの検証が行われる。

【0018】この際、クライアントC側から所望のシステムのデバッグ情報を取得したい旨の信号が送出されると、仮想サーバVSを介してこの所望のシステムのデバッグ情報にアクセス可能なシステム対応サーバSSと自動的に接続される。そのために、クライアントCは、仮想サーバVSおよびシステム対応サーバSSへのアクセスインタフェースとしての機能を果たすライブラリALを保有している。そして、このシステム対応サーバSSとクライアントCとの間でデバッグ情報の要求およびその要求に対する応答がなされ、デバッグ情報が供給される仕組みになっている。そして、一連のデバッグ情報が供給されると、クライアントCはシステム対応サーバSSに対して回線切断要求信号を発行する。一方、システム対応サーバSSは、クライアント側から要求された作業内容を要約したログを仮想サーバVSへ格納すべく、仮想サーバVSにログ格納を依頼する信号を送出する。

【0019】仮想サーバVSは、クライアントCからの接続要求を受け付け、システム対応サーバSSのデモンプロセス（例えば、UNIX上のinetdプロセス）に働きかけてクライアントCとシステム対応サーバSSとを接続する接続用プロセスを生成し、各システム対応サーバとの通信を行う。そして、クライアントCとシステム対応サーバSSとが接続され、デバッグ情報がクライアントCに供給された後、クライアントCからの回線切断要求を受け付け、その旨のメッセージ信号をシステム対応サーバSSに送信し、回線を切断する。このように、仮想サーバVSは中継装置としての機能を有する。

【0020】監視サーバWSは、各システム対応サーバSSのプロセスを監視し、仮想サーバVSに監視結果を随時通知する。これを受けて仮想サーバVSは、システムの負荷分散が図られるようにシステム対応サーバSSの振り分けを行ってクライアントCとシステム対応サーバSSとの接続を行う。また、監視サーバWSは、仮想サーバVSのプロセスについても監視を行っており、仮想サーバVSのプロセスに異常が発見された場合、プロ

セスを再起動し、仮想サーバVSに接続中の全接続プロセスを再構築するようになっている。

【0021】1. 3. システム対応サーバの構成
システム対応サーバSSとして、図1に示したように、システムα用デバッグ情報にアクセス可能なシステムα用サーバSSα、システムβ用デバッグ情報にアクセス可能なシステムβ用サーバSSβといった具合に1種類のデバッグ情報にのみアクセス可能なタイプがある。また、図2に示したように、システムγ用デバッグ情報およびシステムθ用デバッグ情報のいずれかにアクセス可能なシステムγθ用サーバSSγθといった具合に複数種類のデバッグ情報のうちの1つにアクセス可能なタイプもある。そして、これらのタイプのサーバが適宜分散して配置されている。さらに、ネットワークを介して外部からの接続を可能とすべく、デーモンプロセスが起動されている。

【0022】このシステム対応サーバSSは、膨大な交換機用のプログラムを保有している。つまり、ソースプログラム、オブジェクトモジュール、実行モジュールを含めた膨大な数のファイルを記憶装置内に記憶している（これらのファイルの詳細については後述する）。そして、クライアントCからのシンボル変換要求信号を受信してシンボル変換処理を行い、その結果、シンボル情報をクライアントCに送信する。

【0023】また、システム対応サーバSSは、クライアントCからのデバッグ情報取得要求を受けて、汎用フォーマットであるELF(Executable and Linkable Format)/DWARF(Debug With Arbitrary Record Format)に対応した構成要素へのアクセスがなされたものとみなし、内部ではELF/DWARF以外のファイルフォーマットに対応した情報要素へアクセスし、シンボル情報を抽出した上でELF/DWARF形式に変換してクライアントCに結果を返す。

【0024】1. 3. 1. システム対応サーバで取り扱われるファイルフォーマットとデバッグフォーマット
システム対応サーバSSは、既に説明したようにソースファイルや実行形式ファイルを含む交換機用のプログラムに関連する一連の膨大なファイルを保有し、ユーザが指定したプログラムに対するデバッグ結果としてのデバッグ情報をクライアント側に供給する。そこで、このファイルやデバッグ情報のフォーマットについて説明する。

【0025】ファイルフォーマットは、OSがプログラムを実行するために規定された実行形式ファイルの構成を定めたものである。また、デバッグフォーマットは、コンパイラがソースファイルと実行形式ファイルの要素間（ソースの定義と割り付けアドレス等）の対応を生成した情報一式の構成を定めたものであり、主としてデバッガがソースレベルのデバッグを行う際に使用される。ファイルフォーマットとデバッグフォーマットとは、プ

ログラムの構成において対で使用されるものである。

【0026】図8は、このファイルフォーマットとデバッグフォーマットとの関係について示している。同図(a)に示したように、ソースファイル1をコンパイラ2によってコンパイルすることにより、ファイルフォーマットに準拠した実行形式ファイル3が生成される。一方、コンパイル時にデバッグ情報を生成するよう指示が与えられると、デバッグフォーマットに準拠したデバッグ情報4が実行形式ファイル3の一部として（これをセッションという）付加される。

【0027】同図(b)に示したように、ファイルフォーマットとデバッグフォーマットとの対応関係はCPUごとに異なったものとなっている。すなわち、CPUαを搭載したプラットフォームでは、ファイルフォーマット、デバッグフォーマットともにSYSROFが用いられる。また、CPUβを搭載したプラットフォームでは、ファイルフォーマットとしてELF、デバッグフォーマットとしてEWSが用いられ、CPUγを搭載したプラットフォームではファイルフォーマットとしてELF、デバッグフォーマットとしてDWARFが用いられる。

【0028】図5は、これら各種フォーマットを利用するためのシステム対応シンボリックサーバSSの機能構成について示したものである。このように、システム対応シンボリックサーバSSは、クライアントCからのデバッグ情報取得要求に応じて、CPUα用OS個別部Eα、CPUβ用OS個別部Eβ、CPUγ用OS個別部Eγのいずれかを介して、ファイルフォーマットアクセスモジュールFAM、デバッグフォーマットアクセスモジュールDAMにアクセスして必要な情報を取得し、クライアント側に供給する。

【0029】すなわち、クライアントC側からCPUα用OSに対するデバッグ情報の取得要求が発行されると、システム対応サーバSSはCPUα用OS個別部Eαを動作させて、ファイルフォーマットアクセスモジュールFAMおよびデバッグフォーマットアクセスモジュールDAMとして、ともにSYSROF形式のデータにアクセスする。

【0030】同様に、クライアントC側からCPUβ用OSに対するデバッグ情報の取得要求が発行されると、システム対応サーバSSはCPUβ用OS個別部Eβを動作させて、ファイルフォーマットアクセスモジュールFAMとして、ELF形式のデータに、デバッグフォーマットアクセスモジュールDAMとして、EWS形式のデータにアクセスする。

【0031】さらに、クライアントC側からCPUγ用OSに対するデバッグ情報の取得要求が発行されると、システム対応サーバSSはCPUγ用OS個別部Eγを動作させて、ファイルフォーマットアクセスモジュールFAMとして、ELF形式のデータに、デバッグフォー

マッドアクセスモジュールDAMとして、DWARF形式のデータにアクセスする。

【0032】そして、システム対応サーバSSは、このようにアクセスしたデータの中からシンボル変換を行うために必要なデータを抽出してシンボル情報を作成し、クライアントC側に供給する。

【0033】2. 実施形態の動作

2. 1. 基本動作シーケンス

図3は、このようなデバッグシステムにおける基本的な動作シーケンスについて示している。まず、クライアントCのユーザからデバッグ開始指令が与えられると、クライアントCから仮想サーバVSに向けて回線の接続を要求する旨の信号（初期接続要求信号）が送出される

（S1）。仮想サーバVSは各システム対応サーバSSに接続しているユーザ数やプロセスの状況を判断して、回線の接続を承認する旨の信号をクライアントCに送出する（S2）。

【0034】次に、クライアントCからデバッグ対象のシステム名（ α , β , γ , ...）が指定される（S3）。これを受けて、仮想サーバVSは指定されたシステムのデバッグ情報を保有するシステム対応サーバSSが起動されているか否かの確認作業を行う（S4）。指定されたシステムのデバッグ情報を保有するサーバSSの起動が確認されると（S5）、その旨のメッセージ信号とともにこのシステム対応サーバSSの名称と通信ポートとが指定される（S6）。

【0035】クライアントCからデバッグ対象のプログラムに対するシンボル変換を要求する旨の信号が送出されると（S7）、これを受けたシステム対応サーバSSからシンボル情報がクライアントCに返送される（S8）。そして、このステップS7とステップS8の一連の動作に関するログ情報が仮想サーバVSにストックされる（S9）。ソースプログラムのデバッグ過程において、上記ステップS7～S9の動作が繰り返し行われ、一連のデバッグ情報がクライアントCに供給される。

【0036】一通りユーザの所望するデバッグ情報が取得されると、クライアントCから回線の切断要求が発行される（S10）。システム対応サーバSSから回線の切断要求が承認されると（S11）、一連のデバッグ情報取得シーケンスが終了する。これらの動作シーケンスに対するログ情報は仮想サーバVSにストックされる（S12）。

【0037】2. 2. ユーザ側の操作を中心とした動作シーケンス

今度は視点を変えて、デバッグ作業時におけるユーザ側の操作を中心とした動作シーケンスについて、図4を用いて説明する。同図は、クライアントC側のソースレベルデバッガの操作画面におけるユーザ側のオペレーション、仮想サーバVS、システム α 用シンボリック対応サーバSS α 、ソースレベルデバッガ実行部の取り扱い対

象となり、システム α をターゲットとしてコンパイルされたシステム α 用ロードモジュールLM α に関し、これら相互間のやり取りを示したシーケンス図である。

【0038】まず、クライアントC側からシンボリック対応サーバSS α の利用を予約するための信号が送出される（S20）。これを受けて、仮想サーバVSは新たにシンボリック対応サーバSS α の回線接続用のプロセスを起動して回線を接続するよう指示を与える（S21）。そして、仮想サーバVSは、ネットワークからのステータスを元にシンボリックサーバSS α の接続情報を返却して（S22）、クライアントCとシンボリック対応サーバSS α とが接続される。このようにして、一旦、ターゲットデバッグマシンであるシンボリック対応サーバSS α にクライアントCが接続されると、以後の動作では仮想サーバVSを介さず両者の間で直接信号やデータのやり取りが行われる。

【0039】次に、クライアントCからシンボリック対応サーバSS α 内のディスクに記憶され、デバッグ情報を含む多数のファイル（構成ファイルという）の一覧の取得を要求する信号が送出される（S23）。構成ファイルの一覧が取得できたら、クライアントCのユーザは、この構成ファイルの中から所望のソースファイルをオープンし（S24）、ブレークポイントを設定する（S25）。

【0040】そして、このソースファイルの行番号からマイクロプロセッサが実行しうる機械語のアドレスを取得するための行番号アドレス変換指令が、システム α 用シンボリックサーバSS α に向けて発行される（S26）。シンボリックサーバSS α は、ロードモジュールLM α の中から行番号アドレス変換情報を抽出して、クライアントCにアドレス情報を返却する（S27）。なお、この「行番号アドレス変換」の仕組みについては後述する。

【0041】続いて、ユーザの設定したブレークポイントに関する情報が、ロードモジュールLM α に対して設定され、プログラムの先頭からブレークポイントに至るまで実行するよう指示が与えられる（S29）。そして、ブレークポイントまでプログラムが実行されたらデバッガに制御を戻してプログラムの実行を停止し、その旨がクライアントC側に通知される（S30）。このように、通常にプログラムを実行させ、ブレークポイントに到達した時点でデバッガに制御を移行させる方法は、ブレークポイントで指定した番地のマシン語をイリーガルファンクション（IF）と呼ばれる未定義命令に置き換え、このIFを実行することで、CPU Haltを発生させてデバッガに制御を移し、IFを元の命令に書き戻して1ステップ実行させ、またIFを書き込むという動作によって実現される。

【0042】さらに、ユーザはデバッガ操作画面上でソースプログラム内の変数を参照し（S31）、変数アド

レス変換指令をシステムα用サーバSSαに与える（S32）。システムα用サーバSSαは、ロードモジュールLMαから変数アドレス情報を抽出して、ソースプログラム内で参照された変数のアドレスをクライアントC側に返却する（S33）。

【0043】次に、クライアントC側からこの変数の値を参照するようロードモジュールLMαに指令を与える（S34）。すると、ロードモジュールLMαからこの変数の値が返却される（S35）。

【0044】続いて、クライアントCは、ソースレベルデバッガの操作画面において、1ステップずつプログラムをトレースして実行させる指令を与える（S36）。そして、この指令がロードモジュールLMα側に伝送され、1ステップずつプログラムが実行される（S37）。

【0045】その他、種々のデバッグのための操作がデバッガ操作画面からユーザよりなされて（S38）、クライアント側の一連のデバッグ作業が終了する（S39）。そして、シンボリックサーバSSαに対する利用を終了する旨の信号が仮想サーバVSに送信され（S40）、クライアントCとの回線を切断するようシンボリックサーバ側に指示が与えられる（S41）。最後に仮想サーバVSからクライアントC側へシンボリックサーバSSαとの回線が切断され利用が終了した旨の通知がなされる（S42）。

【0046】2. 3. システム対応サーバにおける動作シーケンス

続いて、システム対応サーバSSにおける動作シーケンスについて、特にシステムごとに異なるデバッグ情報の構成の差異を隠蔽するための仕組みに焦点をあてて説明する。

【0047】システム対応サーバSSは、各種ファイルフォーマットやデバッグフォーマットの情報要素を、共通フォーマットであるELFファイルフォーマットとDWARFデバッグフォーマットの情報要素を用いてアクセスし、クライアント側にデバッグ情報を供給する。

【0048】そこで、まず、このELF、DWARFにおける各情報要素間の依存関係について説明する。図9は、ELFファイルフォーマットの構成を示している。このELFファイルフォーマットは、ELFヘッダEL1と、プログラムヘッダEL2と、セクションSEC1、セクションSEC2、・・・、セクションSECn、・・・からなるセクションと、これらのセクションに関する情報を集約したセクションヘッダテーブルEL4とからなる。

【0049】図10は、ELFファイルフォーマットに関し、これら各情報要素の依存関係について示したものである。このファイルフォーマット内には、オブジェクトファイルの種類やプロセッサの種類等を規定したELFヘッダEL1、実行形式ファイルにおけるセグメント

に関する情報等が記録されたプログラムヘッダテーブルEL2、およびセクション名文字列テーブルEL3、セクションヘッダテーブルEL4が格納されている。EL5～EL31は、ELFファイルフォーマットとして種々の拡張子を有するファイルを示している。そして、図9に示したセクション1～セクションn、・・・のそれぞれに対応して、EL5～EL31までの各セクションが格納されている。

【0050】図11は、DWARFファイルフォーマットにおける各情報要素の依存関係について示したものである。このファイルフォーマット内には、.debugセクションEL9にリンクする形でユニット情報DW1、ソースファイル情報DW2、・・・が存在し、以下インライン関数情報DW35に至るまで各種情報が細分化されて格納され、シンボリックデバッグ情報が提供される

（DW1～DW35はそれぞれDWARFデバッグ情報である）。このように、デバッグ情報は実行形式ファイルの一部として実装されるため、デバッグ情報へアクセスするためには実行形式ファイルの先頭から順次、依存関係を手繰っていくことが必要である。

【0051】そこで、図4のステップS26に示した「行番号アドレス変換」において、シンボリックサーバSSがクライアントCから要求を受けた後の内部動作を、既に図5において説明した対応するCPUごとに異なる個別部の動作を中心に説明する。

【0052】2. 3. 1. 行番号アドレス変換の際のシステム対応サーバ側の動作

図6は、上記行番号アドレス変換の詳細を示したフローチャートである。まず、シンボリックサーバSSから行番号アドレス変換の指令がCPUγに対応したシンボリックサーバ個別部Eγに通知される（S50）。この指令を受けたシンボリックサーバ個別部Eγは、ELFファイルフォーマットアクセス部（単にファイルフォーマットアクセス部という）FAにデバッグセクションの取得を要求する（S51）。そしてファイルフォーマットアクセス部FAは、ELFファイル内のELFヘッダEL1を取得する（S52）。

【0053】次に、ファイルフォーマットアクセス部FAは、セクション名文字列テーブルを抽出する動作を行う（S53）。この際、ファイルフォーマットアクセス部FAは、ELFファイルの各構成要素の中からセクション名文字列テーブルEL3を取得する（S54）。さらに、ELFファイルフォーマットアクセス部FAはデバッグセクション名取得のための要求をDWARFデバッグフォーマットアクセス部（単にデバッグフォーマットアクセス部という）DAに与える（S55）。

【0054】続いて、ファイルフォーマットアクセス部FAは、ELFファイルの中の.debugセクションを抽出する動作に移行する（S56）。この過程でファイルフォーマットアクセス部FAは、実際にセクションヘッダ

テーブルEL14を取得する(S57)。

【0055】次に、ファイルフォーマットアクセス部FAは、ELFファイルの中の.debugセクションエントリを抽出する動作に移行する(S58)。この過程で、.debugセクションを取得する(S59)。

【0056】そして、シンボリックサーバ個別部E_γからデバッグフォーマットアクセス部DAに向けて、ユニット情報を取得するための要求が発行される(S60)。これを受けて、デバッグフォーマットアクセス部DAは、DWARFデバッグ情報内のユニット情報を取得する(S61)。そして、デバッグフォーマットアクセス部DAは、このユニット情報内に記録された情報要素を抽出する(S62)。

【0057】続いて、シンボリックサーバ個別部E_δから行番号に対応するアドレスを取得するための要求がデバッグフォーマットアクセス部DAに向けて発行される(S63)。これを受けて、デバッグフォーマットアクセス部DAは、DWARFデバッグ情報内からソースファイル情報DW2を取得し、このソースファイル情報DW2内のデータを参照して行番号に対応したアドレスを取得する(S65)。

【0058】このように、シンボリックサーバは、ユーザからのシンボル変換要求(この例では、「行番号アドレス変換」)を受け付け、この要求を自身が解析可能なファイルフォーマットとデバッグフォーマットのアクセス部DAに向けてELF/DWARF情報要素と見立てた要求(この例では、「デバッグセクション取得」, 「ユニット情報取得」, 「行番号対応アドレス取得」)に変換して内部的なシンボル情報の抽出処理を行い、共通フォーマットであるELF/DWARF形式に変換してクライアントC側に供給する。以上のようにして、システム対応シンボリックサーバSSはCPUごとに異なるデバッグ情報の差異を隠蔽する機能を有している。

【0059】2.3.2. 未定義情報に対する要求が発行された場合のシステム対応サーバ側の動作
一般に、各ファイルフォーマット/デバッグフォーマットは、それぞれに相関がないため、あるフォーマットに規定されている情報要素が他のフォーマットでは規定されていない場合がある。そこで、これらの齟齬を吸収するために、シンボリックサーバSSは各フォーマットアクセス部に規定されていない情報要素へのアクセスが発生した場合に、解析を行うことなく既定値を返却する機構を有する。

【0060】図7は、ファイルフォーマットおよびデバッグフォーマット内に定義されていない情報に対するアクセスがなされた場合の動作について示している。まず、シンボリックサーバSSはユーザ側からの要求に対応する情報がファイルフォーマットおよびデバッグフォーマット内に定義されているか否かの問い合わせを行う(S70)。これを受けてシンボリックサーバ個別部E

は、ファイルフォーマットアクセス部に問い合わせの処理を行う(S71)。そして、ファイルフォーマットアクセス部は、ファイルフォーマットA対応の未定義情報要素にアクセスして、未定義情報要素であるか否かの確認を行う(S72)。

【0061】続いて、シンボリックサーバ個別部Eは、デバッグフォーマットアクセス部に問い合わせの処理を行う(S73)。これを受けて、デバッグフォーマットアクセス部は、デバッグフォーマットA対応の未定義情報要素にアクセスして、未定義情報要素であるか否かの確認を行う(S74)。このように、ファイルフォーマットおよびデバッグフォーマット内に定義されていない情報に対するアクセスがなされた場合には、所定の既定値をクライアント側に返すことで対応する。

【0062】2.4. ローカル変数の型名取得の際の動作

それでは、実際に図12に示したC言語によるソースプログラム(ファイル名test.c)のローカル変数iの型名取得の際の動作を図13~図15を用いて説明する。図13は、システム対応サーバSSにおいて、クライアントCからのデバッグ情報取得要求を受け付ける際の動作について示したものである。

【0063】まず、クライアントCからこのソースプログラムtest.cのmain関数内のローカル変数iの型を解決すべき旨の要求が発行される(S80)。これを受けてシステム対応サーバSSはデバッグ情報アクセス部DAにユニット情報(ソースファイル情報)取得のための要求を発行する(S81)。そして、デバッグ情報アクセス部DAからユニット情報が返却される(S82)。

【0064】次に、システム対応サーバSSはPROC情報(グローバル関数情報)取得のための要求をデバッグ情報アクセス部DAに向けて発行する(S83)。これを受けてPROC情報がデバッグ情報アクセス部DAからシステム対応サーバSSに返却される(S84)。さらに、変数情報としてのローカル変数情報の取得要求に応答して、ローカル変数情報がデバッグ情報アクセス部DAからシステム対応サーバSSに返却される(S85, S86)。このようにして、変数iの型が整数型のintである旨の通知がシステム対応サーバSSからクライアントCに向けてなされる(S87)。

【0065】2.4.1. DWARFにおけるデバッグ情報取得の際の動作

このようなソースプログラムの型解決処理において、デバッグDGとデバッグ情報アクセス部DAとの間では図14に示したような動作シーケンスが実行される。同図では、ユニット情報(ソースファイル)、PROC情報(グローバル関数情報)、変数情報(ローカル変数情報)のそれぞれについて、デバッグ情報アクセス部DAから返却されるデータの構成をシーケンス図中に明示している。

【0066】これらDWARFにおけるデバッグ情報（シンボルテーブル）の一要素であるユニット情報、PROC情報、変数情報のそれぞれの詳細なデータ構成については、図15（a）～（c）に示している。なお、これら情報要素のDWARFデータの体系における位置付けについては、既に説明した図11に示した（ユニット情報DW1，グローバル関数情報DW9，ローカル関数情報DW14）通りである。

【0067】図15（a）に示したように、ユニット情報は、上位バイトから下位バイトに向かって、ユニット名（n+1）バイト、使用言語4バイト、ユニット開始アドレス4バイト、ユニット終了アドレス4バイト、行番号情報オフセット4バイト、コンパイルディレクトリ（n+1）バイト、作者名（n+1）バイトの順で配置された合計3n+19バイトのデータ列DL1によって構成される。同様にPROC情報、および変数情報の場合もそれぞれ図15（b），（c）の表に示した属性要素の順で配置されたデータ列DL2，DL3によって構成されている。

【0068】図14に示したユニット情報取得要求（S81）とユニット情報返却（S82）における処理では、デバッグ情報アクセス部DAからデータ列DL1からなるユニット情報がデバッガDGに返却される。従って、デバッガDG側ではデータ列DL1の上位（n+1）バイトに記録された内容を読み取り、test.cというソースファイル名を取得する。

【0069】次に、PROC情報取得要求（S83）とPROC情報返却（S84）における処理では、デバッグ情報アクセス部DAからデータ列DL2からなるユニット情報がデバッガDGに返却される。従って、デバッガDG側ではデータ列DL2の上位（n+1）バイトに記録された内容を読み取り、mainというグローバル関数名を取得する。

【0070】さらに変数情報取得要求（S85）と変数情報返却（S86）における処理では、デバッグ情報アクセス部DAからデータ列DL3からなるユニット情報がデバッガDGに返却される。従って、デバッガDG側ではデータ列DL3の上位（n+1）バイトに記録された内容を読み取り、intというローカル変数の型名を取得し、「ローカル変数iの型がintである」というシンボリック情報を取得することができる。

【0071】3. 実施形態の効果

（1）リモートログインせずにローカル端末でデバッグ作業をすることができ、仮想サーバがターゲットシステム専用サーバに振り替えるところから、マシンの負荷分散を図ることができる。そして、この仮想サーバの振り替え機能により、クライアントは多くのサーバを意識することなく、仮想サーバだけを認識するだけでシンボリック情報を取得してデバッグ作業を行うことができる。

（2）クライアント側のデバッグ用ツールからみると、

ある程度デバッグ情報が異なるものに対しても、同じインタフェースでデバッグ情報にアクセスが可能となり、プロセッサごとに異なるデバッグ情報に対応して、個別にデバッグ用ツールを開発する必要がない。

（3）さらに、各種デバッグフォーマットに対応する部分（システム対応サーバ側の個別部）をモジュール化することで、対応するデバッグフォーマットの種類を随時増やすことができる。

【0072】4. 変形例

本発明は、上記実施形態に限定されるものではなく、例えば以下のように種々の変形が可能である。

（1）本実施形態においては、デバッグ情報を生成するコンパイラとして、CコンパイラまたはC++コンパイラを念頭において説明したが、個別のアクセスルーチンを追加することで、その他のプログラム言語（例えばJava等）のコンパイラにも対応可能である。

（2）本実施形態においては、仮想サーバが1台の場合を例に挙げたが、多数のクライアントからの処理が集中するような場合には、仮想サーバとしての機能を有するミラーサーバを設けて、このミラーサーバに適宜処理を分散させることも可能である。

【0073】

【発明の効果】以上詳細に説明したように、この発明によれば、プログラムのデバッグ作業に際し、リモートログインせずにローカル端末から共通のインタフェースでシンボル情報を取得することができる。

【図面の簡単な説明】

【図1】本実施形態にかかるデバッグシステムの概要を示した構成図である。

【図2】同上デバッグシステムの具体的なシステム構成について示したものである。

【図3】クライアント、仮想サーバ、各システム対応サーバ間の動作シーケンスについて示したものである。

【図4】同上デバッグシステムを用いてデバッグ作業を行う際のユーザ側を中心とした処理の流れについて示したものである。

【図5】システム対応シンボリックサーバの機能構成について示したものである。

【図6】ELF/DWARFを用いて行番号アドレス変換が行われる際のシーケンスを示したものである。

【図7】ファイルフォーマットやデバッグフォーマットに定義されていない情報要素にアクセスする際の機構について示したものである。

【図8】ファイルフォーマットとデバッグフォーマットとの関係について示したもので、（a）はデバッグ作業面からみた双方の関係について示し、（b）は具体的な各フォーマットの相互間の関係を示したものである。

【図9】ELFファイルフォーマットの構成を示したものである。

【図10】ELFファイルフォーマットにおける各情報

要素間の依存関係について示したものである。

【図11】DWARFデバッグフォーマットにおける各情報要素間の依存関係について示したものである。

【図12】ソースプログラムの一例を挙げ、特にそのソースプログラムに記述されたローカル変数について示したものである。

【図13】クライアントからシステム対応サーバにデバッグ情報取得要求が発行されてから、実際にデバッグ情報が返送されるまでのシーケンスの一例を示したものである。

【図14】DWARFにおけるシンボリック情報取得の際の状況を示したものである。

【図15】DWARFにおけるシンボルテーブルの構成を示したもので、(a)はユニット情報を、(b)はグローバル関数情報を、(c)はローカル変数情報を示している。

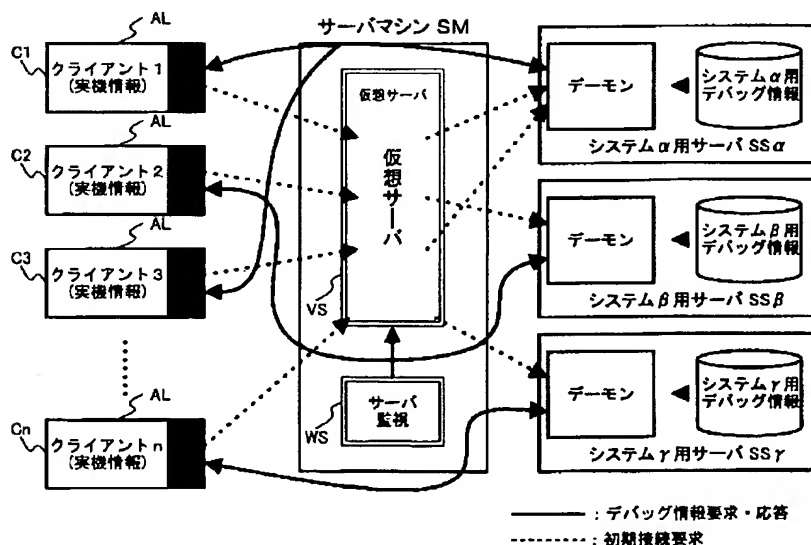
【図16】従来の遠隔デバッグシステムの構成を示したものである。

【符号の説明】

- 1 ソースファイル
- 2 コンパイラ

- 3 実行形式ファイル
- 4 デバッグ情報
- VS 仮想サーバ
- WS 監視用サーバ
- SM サーバマシン
- AL アクセス用ライブラリ
- ENET イーサネット (登録商標)
- FA ファイルフォーマットアクセス部
- DA デバッグフォーマットアクセス部
- SS ($S\alpha$, $S\beta$, $S\gamma$, ...) システム対応 (シンボリック) サーバ
- C ($C1$, $C2$, $C3$, ...) クライアント
- E ($E\alpha$, $E\beta$, $E\gamma$, ...) プロセッサごとの個別部
- LM ($LM\alpha$, $LM\beta$, $LM\gamma$, ...) ロードモジュール
- EL1~EL31 ELFファイルフォーマット構成要素
- DW1~DW35 DWARFファイルフォーマット構成要素

【図1】

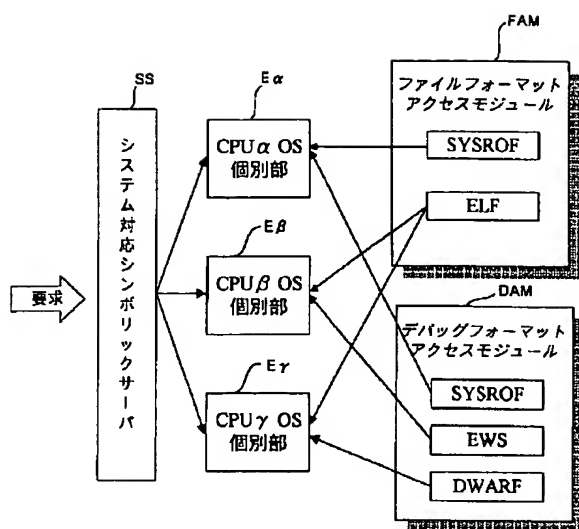


【図4】

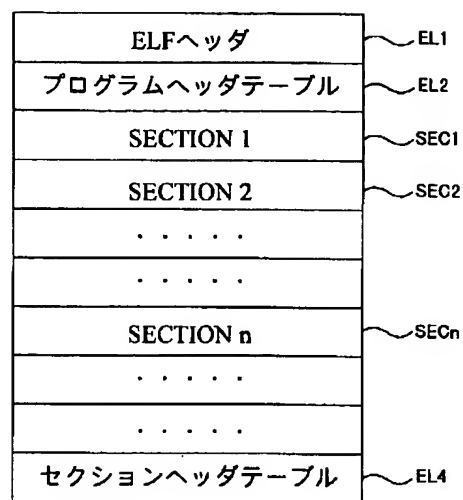
クライアント C	仮想サーバ V S	各システム 対応サーバ S S
デバッグ 開始	接続要求 (S1)	
	接続承認 (S2)	
	デバッグ対象指定 (S3)	各システム対応 サーバの確認 (S4)
	指定 OK (対応サーバと (S6) ポートを指定)	起動確認 (S5)
	シンボル変換要求 (S7)	
		シンボル情報 (S8)
		ログ情報 (S9)
	⋮	⋮
	⋮	⋮
	⋮	⋮
	切断要求 (S10)	
		切断承認 (S11)
デバッグ 終了		ログ情報 (S12)



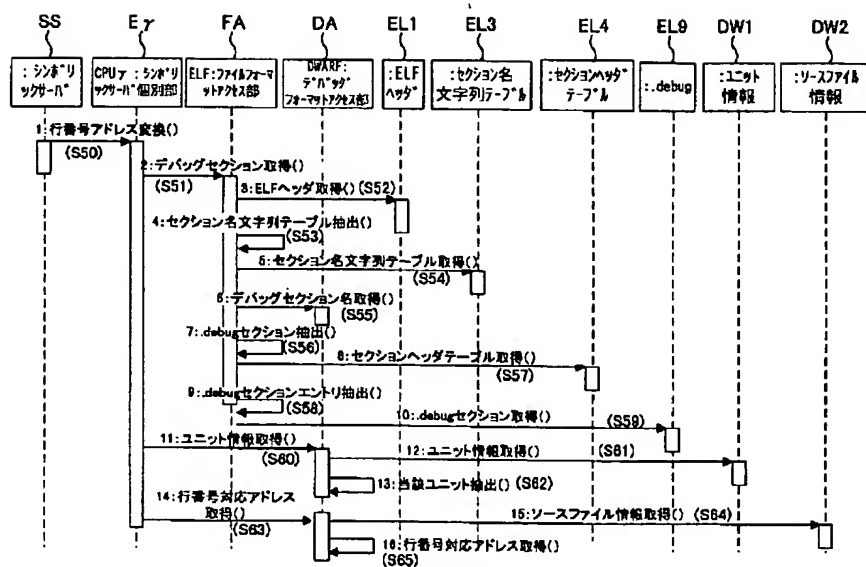
【図5】



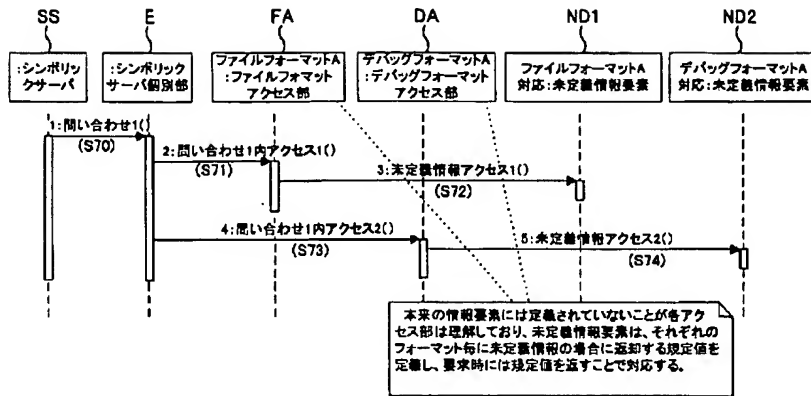
【図9】



【図6】

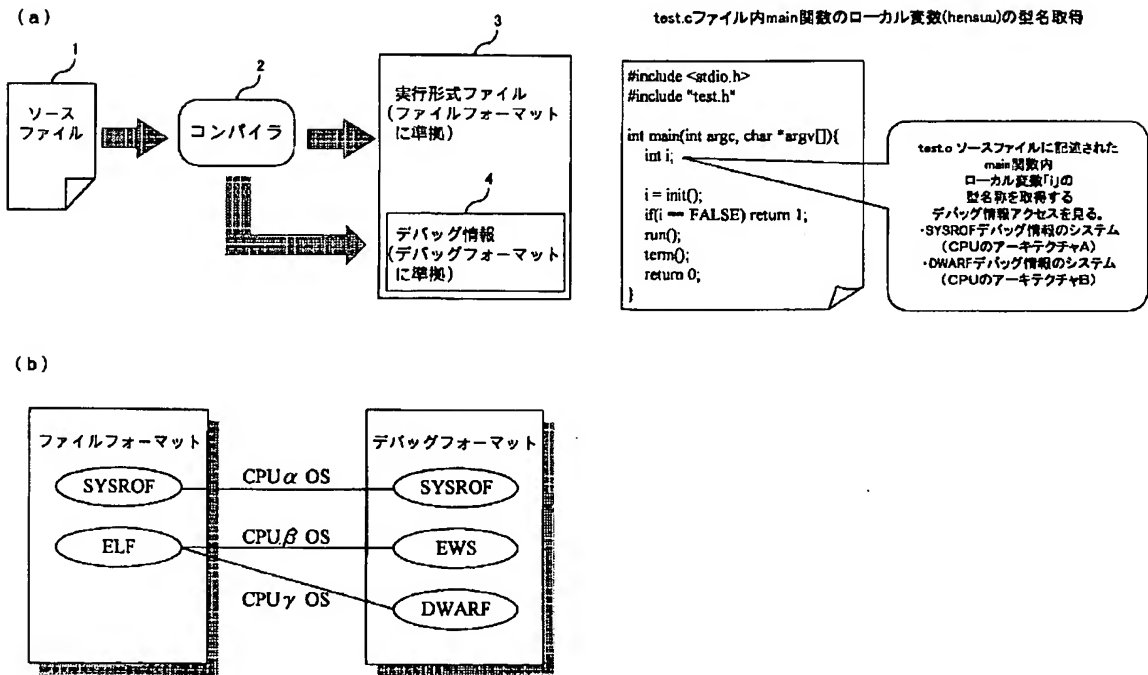


【図 7】

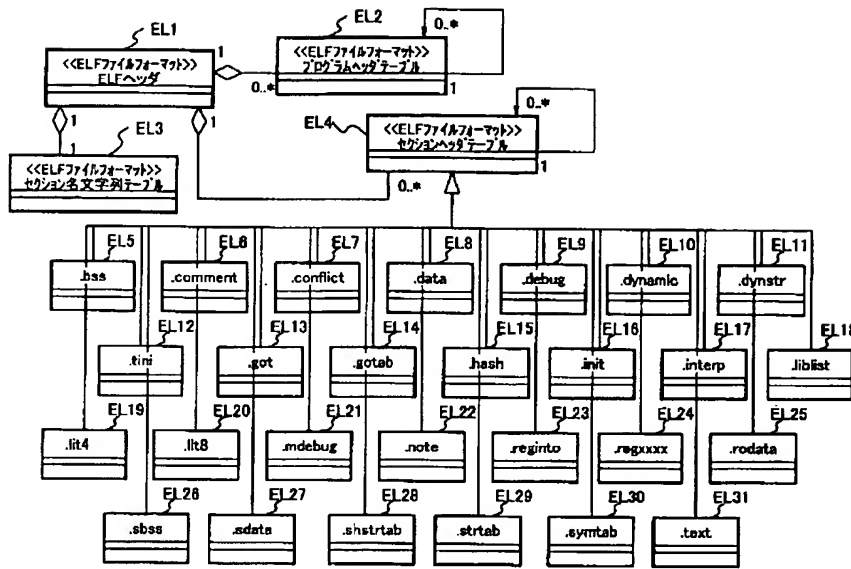


【図 8】

【図 12】



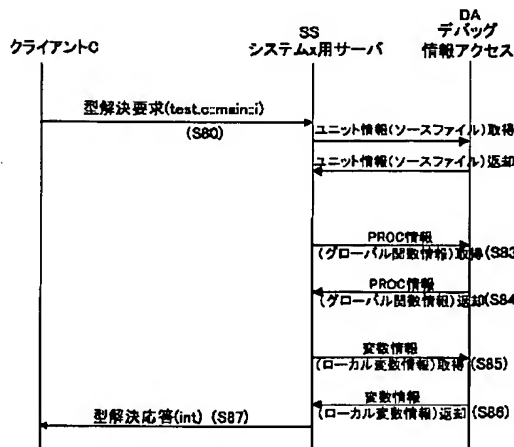
【図10】



【図13】

【図15】

システム対応サーバの受け付けイメージ



(a)

DWARF
ユニット情報(ソースファイル) DW1

DWARF:ユニット情報(ソースファイル)			
	属性	サイズ(byte)	型
1	ユニット名	n+1	string
2	使用言語	4	word
3	ユニット開始アドレス	4	address
4	ユニット終了アドレス	4	address
5	行番号情報オフセット	4	word
6	コンパイルディレクトリ	n+1	string
7	作成者	n+1	string

(b)

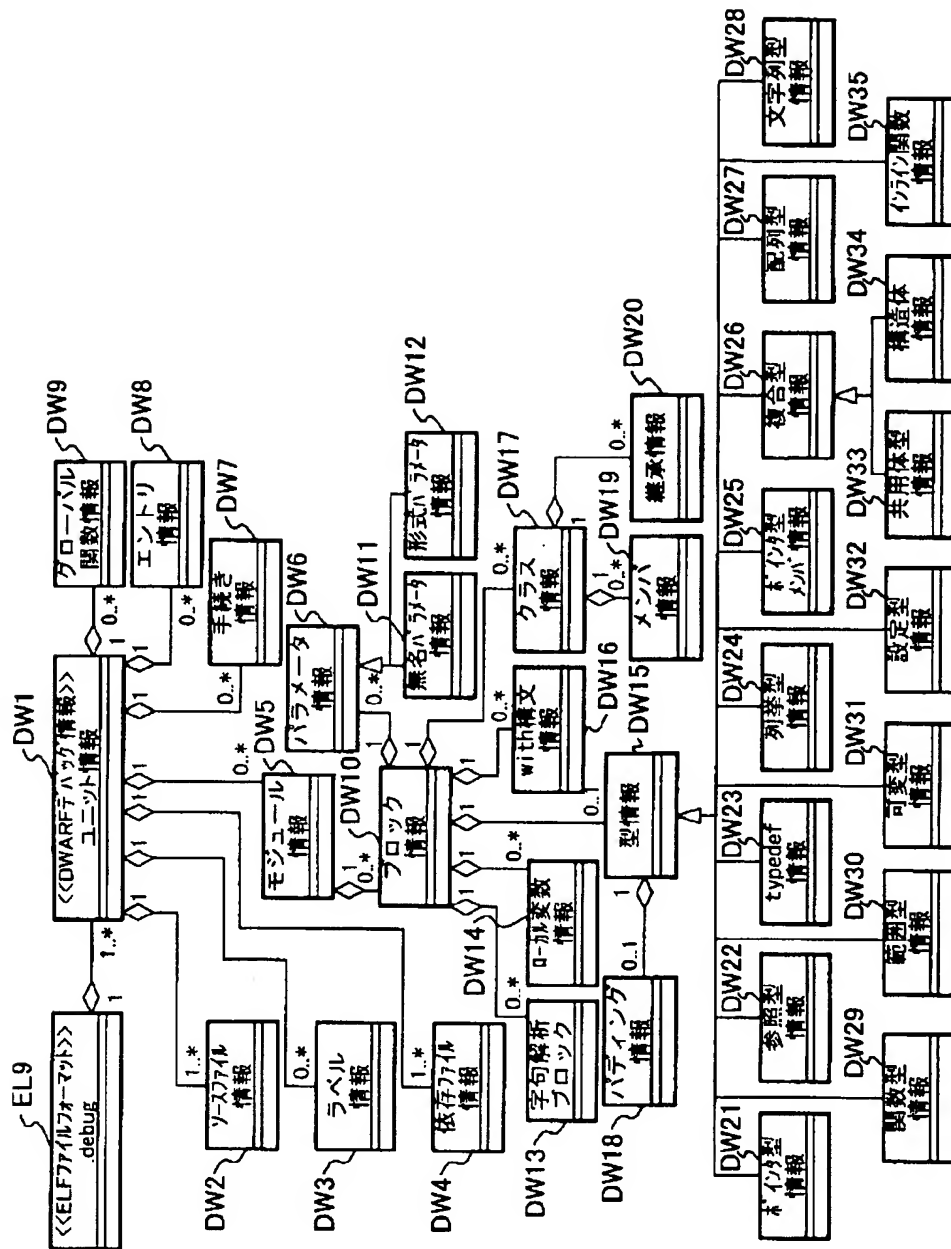
DWARF
PROC情報(グローバル関数情報) DW9

DWARF:PROC情報(グローバル関数情報)			
	属性	サイズ(byte)	型
1	関数名	n+1	string
2	基本型情報	2	word
3	修飾基本型情報		block2
4	ユーザ型情報	4	reference
5	修飾ユーザ型情報		block2
6	関数開始アドレス	4	address
7	関数終了アドレス	4	address

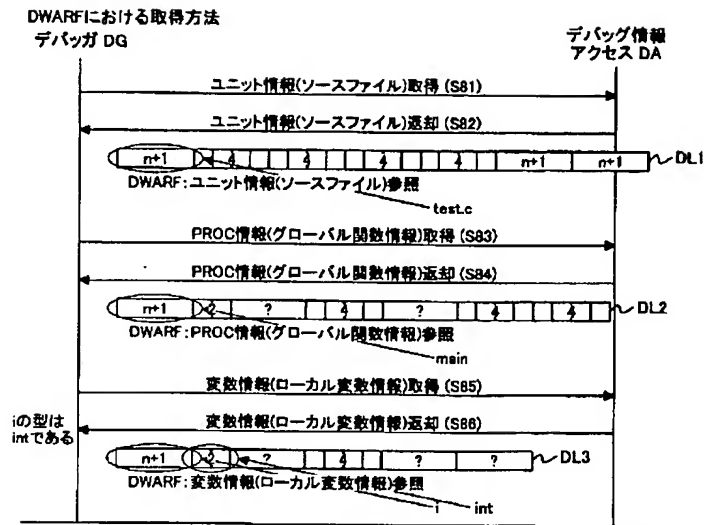
(c)

DWARF
変数情報(ローカル変数情報) DW14

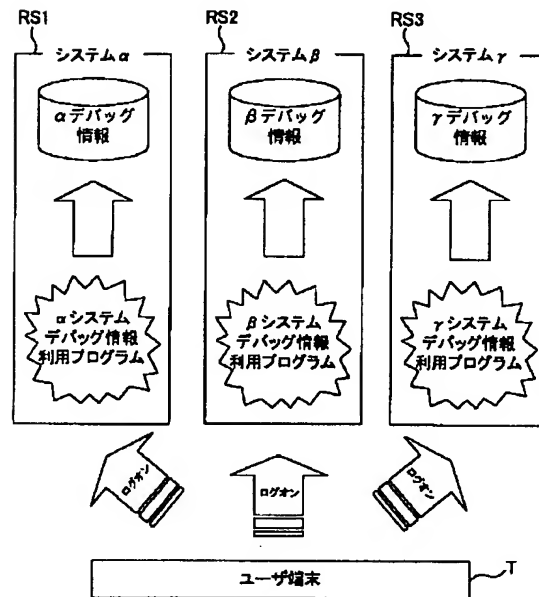
DWARF:変数情報(ローカル変数情報)			
	属性	サイズ(byte)	型
1	変数名	n+1	string
2	基本型情報	2	word
3	修飾基本型情報		block2
4	ユーザ型情報	4	reference
5	修飾ユーザ型情報		block2
6	ローケーション情報		block2



【図14】



【図16】



フロントページの続き

(72)発明者 山田 洋一
東京都港区港南一丁目9番1号 エヌ・テ
ィ・ティ・コミュニケーションウェア株式
会社内

(72)発明者 神人 将彰
東京都港区港南一丁目9番1号 エヌ・テ
ィ・ティ・コミュニケーションウェア株式
会社内

(72)発明者 楯 武士

東京都港区港南一丁目9番1号 エヌ・テ
ィ・ティ・コミュニケーションウェア株式
会社内

Fターム(参考) 5B042 GA02 GA12 GC08 HH04 HH41
5B045 GG01 JJ49
5B085 AC09 BG07